Scene Text Extraction using Deep Learning

Dissertation submitted to
Visvesvaraya National Institute of Technology, Nagpur
in partial fulfillment of the requirements for the award of
the degree

Master of Technology

in

Computer Science

by

Boya Aarya Devarla (MT23MCS002)

Under the guidance of

Dr. Shital A. Raut

Department of Computer Science and Engineering
Visvesvaraya National Institute of Technology
Nagpur 440 010 (India)

JUNE 2025

Scene Text Extraction using Deep Learning

Dissertation submitted to
Visvesvaraya National Institute of Technology, Nagpur
in partial fulfillment of the requirements for the award of
the degree

Master of Technology

in

Computer Science

by

Boya Aarya Devarla (MT23MCS002)

Under the guidance of

Dr. Shital A. Raut

Department of Computer Science and Engineering
Visvesvaraya National Institute of Technology
Nagpur 440 010 (India)

JUNE 2025

©Visvesvaraya National Institute of Technology (VNIT) 2025

Department of Computer Science and Engineering
Visvesvaraya National Institute of Technology, Nagpur

Declaration

I, **Boya Aarya Devarla**, hereby declare that this dissertation titled **Scene Text Extraction using Deep Learning** is carried out by me in the **Department of Computer Science and Engineering** of Visvesvaraya National Institute of Technology, Nagpur. The work is original and has not been submitted earlier whole or in part for the award of any degree/diploma at this or any other Institution / University.

Boya Aarya Devarla
Date: 12/6/25

Certification

This is to certify that the dissertion titled **Scene Text Extraction using Deep Learning** submitted by **Boya Aarya Devarla** is partial fulfillment of the requirements for the award of the degree the **Master of Technology** in VNIT Nagpur. The work is comprehensive, complete and fit for final evaluation.

12/6/25

Dr. Shital A. Raut
Associate Professor, CSE, VNIT Nagpur

Dr. M. P. Kurhekar
Head, Department of Computer Science and Engineering
VNIT, Nagpur
Date:

विभागप्रमुख
Head
संगणक विज्ञान एवं अभियांत्रिकी विभाग
Dept. of Computer Science & Engg.
वि.रा.रा.प्रौ.सं. नागपुर-१० (भारत)
V.N.I.T., Nagpur-440010. (India)

Department of Computer Science & Engineering
Visvesvaraya National Institute of Technology, Nagpur

NON-PLAGIARISM CERTIFICATE

Certified that M. Tech Dissertation titled **"Scene Text Extraction using Deep Learning"** submitted by **Boya Aarya Devarla**, Enrollment No. **MT23MCS002**, ID No. **30083** has been checked for plagiarism using TURNITIN software and the overlap is found within prescribed limits. A summary of the report is given below.

ORIGINALITY REPORT

11%	9%	7%	5%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

12/6/25

Boya Aarya Devarla
Enrollment No: MT23MCS002

12/6/25

Dr. Shital A. Raut
Associate Professor
Department of Computer Science & Engineering
VNIT, Nagpur

ACKNOWLEDGEMENTS

I want to express my heartfelt gratitude and sincere appreciation to everyone who has overtly or covertly supported and guided me throughout my M.Tech journey.

I find myself fortunate to be guided by **Dr. Shital A. Raut** Madam. Her invaluable expertise and guidance along with his constant encouragement and support in all challenging situations, have been instrumental in shaping this thesis. I will always be grateful to be chosen to take up this research under her guidance.

I would like to heartfully thank the faculty members of the **Department of Computer Science and Engineering** at Visvesvaraya National Institute of Technology Nagpur as they have showered me with knowledge and confidence to deal with any oncoming challenges in my life. I am indebted to them for their teachings and guidance which have broadened my perspective and enhanced my understanding of the subject. My sincere appreciation goes to them for their assistance, collaboration, and for providing the necessary resources for conducting experiments and carrying out this research work. Their contributions have been significant in the successful completion of this thesis.

I would like to extend my gratefulness to the Government of India, for their Central Government Health Scheme for the students of VNIT has helped me get through very tough times.

I am indebted to my family and friends for they have always been there for me, through the highs and the lows of this journey. Their love and motivation have been the driving force behind my accomplishments. Thank you all for being a part of this journey.

Boya Aarya Devarla

ABSTRACT

Scene Text extraction is a fundamental prerequisite for automation, decision-making and other real-world applications, especially in a multilingual agnostic environment, as in Indic scripts. It is a process of identifying and transcription of textual content in a real-world photo. So, linguistically diverse and complex scripts of Indic scripts are a challenging requirement for any deep learning model proposed for Indic script recognition, which is the motivation why we chose this topic for our thesis.

The proposed method makes use of state-of-the-art architectures such as UperNet with Swin backbone and DBNet for text detection with a segmentation precision of 86%. For text recognition, we use the TrOCR framework with DeiT/Swin as the encoder and IndicBART as the decoder, which achieves a CRR of over 90% in multiple Indic languages.

Despite these encouraging results, challenges like non-uniform training dataset annotations for multilingual images and the lack of strong native script language models suggest directions for the future. Identifying languages based on scripts became difficult, as Hindi and Marathi (Devanagari Script) or Assamese and Bengali (Eastern Nagri Scripts) is also another challenge for the approach which was proposed.

We would like to explore lightweight models for on-device execution in real-time to improve computational efficiency and inclusion in future work. This work is an important first step towards demonstrating Indic text recognition technology and can serve as an exciting first step to mitigate technological and linguistic disparity in the domain of scene text extraction.

Keywords: Computer Vision, Machine Learning, Transformers, Vision Transformers. Language Models, Scene Text Extraction, Text Detection, Text Recognition, OCR

CONTENTS

CERTIFICATE . . . iii
NON-PLAGIARISM CERTIFICATE . . . iv
ACKNOWLEDGEMENTS . . . v
ABSTRACT . . . vi

1 Introduction **1**
1.1 Motivation . . . 1
1.2 Problem Statement . . . 1
1.3 Objectives . . . 2
1.4 Scope . . . 3
1.5 Thesis Outline . . . 3

2 Literature Review **4**
2.1 Overview . . . 4
2.2 Scene Text Detection . . . 5
2.2.1 History of Early Methods . . . 5
2.2.2 Recent Advancements . . . 7
2.2.3 Datasets . . . 8
2.2.4 Challenges . . . 9
2.2.5 Evaluation Metrics . . . 10
2.3 Scene Text Recognition . . . 11
2.3.1 History of Early Methods . . . 11
2.3.2 Recent Advancements . . . 11
2.3.2.1 CNN-Based Approaches . . . 11
2.3.2.2 Transformer-Based Approaches . . . 12
2.3.2.3 Hybrid and Emerging Approaches . . . 12
2.3.3 Datasets . . . 13
2.3.4 Challenges . . . 15
2.3.5 Evaluation Metrics . . . 16

3 Methodology and Implementation **17**
3.1 System Requirements . . . 17
3.2 Detection . . . 18
3.2.1 Model Architecture . . . 18
3.2.2 Preprocessing . . . 20
3.2.3 Hyperparameter Configuration . . . 20
3.2.4 Loss Function: DBLoss . . . 21
3.2.5 Implementation . . . 21
3.2.6 Training and Evaluation . . . 21
3.3 Recognition . . . 22
3.3.1 Model Architecture . . . 22
3.3.2 Preprocessing . . . 24
3.3.3 Hyperparameters Configuration . . . 25
3.3.4 Implementation . . . 26
3.3.5 Training and Evaluation . . . 26

4 Results and Discussions **27**
4.1 Scene Text Detection . . . 27
4.1.1 Results on TotalText Dataset . . . 28
4.1.2 Results on SynthText Dataset . . . 28
4.1.3 Fine-Tuning with TotalText and IndicSTR12 . . . 29
4.1.4 Discussion . . . 30
4.2 Scene Text Recognition . . . 32
4.2.1 Results . . . 33
4.2.1.1 Good Examples . . . 33
4.2.1.2 Moderate Examples . . . 34
4.2.1.3 Poor Examples . . . 35
4.2.2 Discussion . . . 36

5 Conclusion and Future Work **37**
5.1 Conclusion . . . 37
5.2 Compliance . . . 37
5.3 Limitations . . . 39
5.4 Future Work . . . 39

Bibliography **43**

LIST OF FIGURES

2.1 Illustration of the predominant method for scene text extraction. 5
2.2 Example of Maximal Stable Extreme Region (MSER) used in scene text detection. One of the CCA techniques. [19] 6
2.3 Example of Stroke Width Transform (SWT) used in scene text detection. [25] 6
2.4 Modular Architecture for Scene Text Detection: Backbone, Neck, and Head 8
2.5 Sample Images from the datasets 9
2.6 IndicSTR12 Dataset Example 14

3.1 Proposed Model 19
3.2 TrOCR Architecture 22
3.3 Ideal vs. Actual Workflow when Training Recognition Model due to the Limitation of IndicBART 24

4.1 Graphs for both the backbones Loss vs. Epoch 28
4.2 Loss Comparision between tiny and base variants of Swin Backbone based model 29
4.3 Different Examples compared to their ground truths 30
4.4 More Examples of Scene Text Detection 30
4.5 Example of Inconsistent Annotations 31
4.6 Example of detecting small texts 32

LIST OF TABLES

2.1 Summary of Datasets for Scene Text Detection 9
2.2 Summary of Datasets for Scene Text Recognition 14

3.1 System Specifications 17
3.2 Important Python Packages for Scene Text Extraction 18

4.1 Performance metrics for Swin and ConvNext backbone variants trained on the TotalText dataset. 28
4.2 Performance metrics for Tiny and Base variants trained on the TotalText dataset. 29
4.3 CER and WER for DeIT Encoder for TrOCR 33
4.4 CER and WER for Swin Encoder for TrOCR 33

Chapter 1

INTRODUCTION

1.1 Motivation

India is home to various different languages, which are expressed through different scripts. In this growing digital ecosystem, robust text extraction technologies are essential for applications such as navigation systems, digitalisation of documents, translation, and many more. Scene text extraction, which involves detecting and recognising text from natural images helps is critical for India.

Indic languages have unique challenges due to their visual complexities, characters and overlaps of characters from using the same script. These pose a much more complex space compared to Latin languages such as English. As a result, models trained only for Latin scripts often fail to achieve good results in Indic contexts [22, 21].

Recent advancements in deep learning, particularly in CNN and Transformer-based architectures, have significantly improved vision and language tasks. Making use of these advanced models to create a text extraction pipeline for Indic Text is promising. Such advancements have the potential to preserve the language, improve literacy, and broaden socio-economic benefits.

1.2 Problem Statement

Scene text extraction refers to the identification and transcription of textual content in natural, real-world images, such as signs, product labels, etc. Unlike traditional OCR, which works in a controlled environment, scene text extraction poses unique challenges due to complex backgrounds, different fonts,

distortions, and text orientations. The aim of the project is to develop a robust end-to-end deep learning pipeline that can identify and recognize text from real-world scene images.

1.3 Objectives

The objectives of this project are organized into two primary tasks: scene text detection and scene text recognition.

A. Scene Text Detection Objectives

- Develop a deep learning-based model capable of accurately identifying text regions in complex scene images.
- Use different datasets to test and improve the overall quality of the model, including the use of a variety of Indic languages, fonts, and images.
- Generate high-quality segmentation masks for further recognition processing.
- Evaluate the detection system on standard benchmarks using Precision, Recall and F-Score.

B. Scene Text Recognition Objectives

- Design a recognition model that can transcribe cropped text images into machine-readable sequences with high accuracy, even under noise or partial occlusion.
- Improve robustness of character-level recognition by leveraging sequence-to-sequence learning and contextual language modeling.
- Support multi-lingual or script-agnostic recognition for better adaptability to various real-world settings.
- Evaluate the recognition system on standard benchmarks using CER (Character Error Rate) and WER (Word Error Rate).

1.4 Scope

The scope of this thesis is limited to natural scene images containing textual content. The detection and recognition models employs Vision Transformer as the recent trend shifts towards transformer-based approaches. The system will be trained and evaluated using publicly available datasets. The models will be assessed for their performance using the standard metrics such as Precision, Recall and F-Score for detection and task, and for recognition, we use Character Error Rate (CER) and Word Error Rate(WER).

The study explicitly excludes handwriting recognition and traditional document OCR, maintaining a strict emphasis on the extraction of scene text from natural images.

1.5 Thesis Outline

The thesis is structured as follows:

- **Chapter 1: Introduction** –This chapter provides an overview of the project's background, problem statement, objectives, scope, and structure.
- **Chapter 2: Literature Review** – Surveys prior work in OCR, scene text detection and recognition, and recent developments in deep learning relevant to the task.
- **Chapter 3: Methodology and Implementation** – Details the proposed detection and recognition approaches, architectural choices, training strategies, and implementation details.
- **Chapter 4: Results and Evaluation** – Presents evaluation metrics, experimental outcomes, visualizations, and a comparative performance analysis.
- **Chapter 5: Conclusion and Future Work** – Summarizes findings, discusses limitations, and proposes directions for future research.

Chapter 2

LITERATURE REVIEW

2.1 Overview

Scene text extraction, encompassing both detection and recognition (See fig. 2.1), which involves both detecting and recognizing text in everyday images has become an important problem in the field of computer vision. Whether it's reading street signs, identifying labels on products, or translating billboards, the ability to pull text from real-world environments has huge practical value. Unlike traditional OCR systems that work on controlled environment (such as, scanned documents), Scene Text Extraction operates in far complex conditions like varied lighting conditions, complex or noisy backgrounds, skewed perspectives, and text that may appear in all kinds of fonts, sizes, and orientations [13] [3] [5] [25].

Over the years, the field has evolved from basic, hand-crafted techniques inspired by document analysis to more sophisticated deep learning-based models that combine detection and recognition in a single pipeline. These end-to-end systems have dramatically improved performance, especially when powered by convolutional and recurrent neural networks.

However, when it comes to Indic scripts, such as Devanagari, Tamil, and Bengali, the problem becomes even more complicated. These scripts are not only visually dense and structurally diverse, but they also lack the abundance of high-quality, annotated datasets that exist for English and other Latin scripts [21]. This shortage of data, combined with the unique shapes and ligatures of Indic characters, makes it harder for models trained on standard datasets to generalize effectively. Nonetheless, deep learning techniques particularly

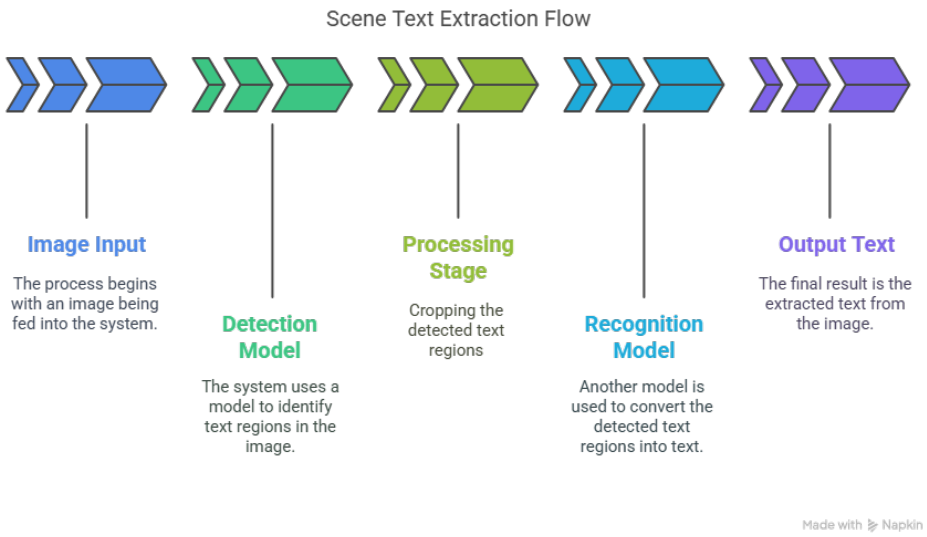

Scene Text Extraction Flow
Image Input
The process begins with an image being fed into the system.
Detection Model
The system uses a model to identify text regions in the image.
Processing Stage
Cropping the detected text regions
Recognition Model
Another model is used to convert the detected text regions into text.
Output Text
The final result is the extracted text from the image.
Made with Napkin

Figure 2.1: Illustration of the predominant method for scene text extraction.

CNNs and RNNs have made meaningful progress by automating feature extraction and improving adaptability across a wider range of scenes and languages [33, 13, 5].

2.2 Scene Text Detection

2.2.1 History of Early Methods

In the early days of scene text detection, researchers adapted document analysis techniques that were designed to work in controlled environments. But with scene text detection, the input is irregular. The techniques typically followed a sequential pipeline that involved preprocessing to enhance the image quality, then using detection to locate the text regions, and recognition to transcribe the text. Techniques relied on hand-crafted features like edge detection, connected component analysis (see Fig.2.2), and stroke width transform (SWT) (see Fig.2.3) [25]. However, these methods struggled with the variability of natural scenes, including uneven lighting, perspective distortions, and complex backgrounds, limiting their effectiveness [33].

Figure 2.2: Example of Maximal Stable Extreme Region (MSER) used in scene text detection. One of the CCA techniques. [19]

Figure 2.3: Example of Stroke Width Transform (SWT) used in scene text detection. [25]

2.2.2 Recent Advancements

In recent years, advances in scene text detection have been accelerated by deep learning and, more particularly, by the use of Convolutional Neural Networks (CNNs). These networks automate feature extraction and improve robustness with respect to noise, illumination variations, and multi-oriented text. They are often categorised as:

- **Regression-Based Methods**: Similar to object detection, we predict the bounding boxes of textual content. Examples include Deep Matching Prior Network (DMPNet) and Rotation Region Proposal Network (RRPN), which handle arbitrary-oriented texts [19].
- **Segmentation-Based Methods**: Similar to Semantic or Instance Segmentations we detect the region were the textual content exists. Examples include Mask R-CNN.
- **Hybrid Methods**: Combining regression and segmentation, methods like LOMO integrate proposal and instance segmentation for improved accuracy on complex text shapes [19].
- **End-to-End Text Spotters**: Frameworks like Mask TextSpotter and FOTS integrate detection and recognition, enhancing efficiency and accuracy for irregular texts [19].

Recent advancements indicate a modular approach for the architecture. It comprises of three key components: ***Backbone, Neck, and Head*** (See Fig.2.4). The **Backbone**, typically a deep convolutional neural network (e.g., ResNet or EfficientNet), is responsible for the extraction of robust feature representations from input images.
The **Neck**, such as Feature Pyramid Networks (FPN), is responsible for handling different scales of texts in the image.
The **Head** is responsible for predicting the textual regions either by generating the bounding box or creating a segmentation mask.

This structured pipeline, inspired by general object detection frameworks, has become a dominant trend, enabling efficient and accurate scene text detection across diverse datasets. [27] [28]

For Indic texts, these methods are being adapted to handle complex scripts, though challenges remain due to limited training data [19].

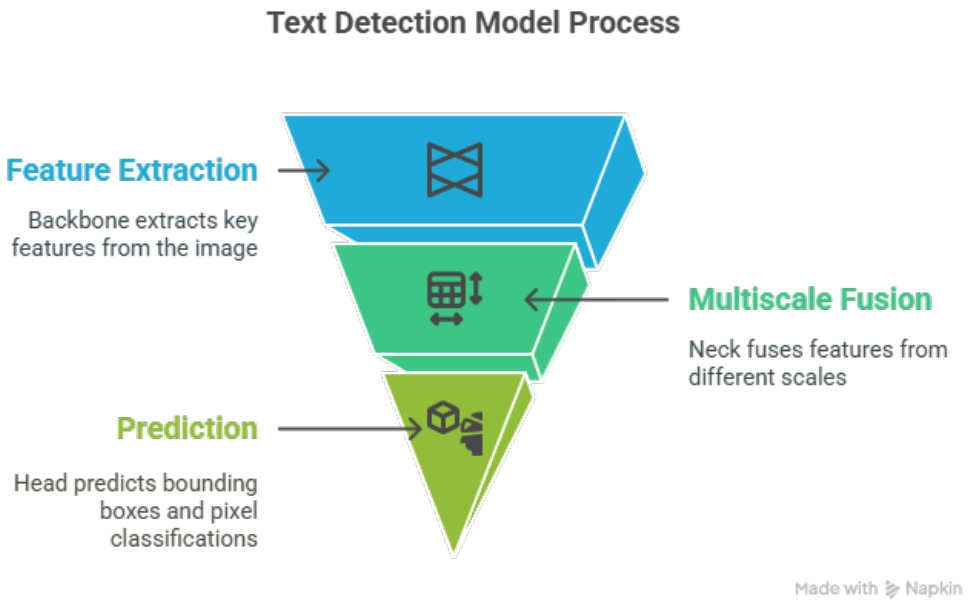


Figure 2.4: Modular Architecture for Scene Text Detection: Backbone, Neck, and Head

2.2.3 Datasets

To evaluate scene text detection algorithms, several benchmark datasets are utilized:

- **ICDAR Series (2003–2019)**: A collection of datasets providing benchmarks for text localization, segmentation, and recognition, with recent versions addressing multilingual and curved text scenarios.
- **MSRA-TD500**: Comprises images containing multi-oriented text in English and Chinese, annotated at the text-line level.
- **COCO-Text**: A large-scale dataset featuring natural images with diverse text instances, including machine-printed and handwritten text.
- **Total-Text and CTW-1500**: Focus on curved text in natural scenes, offering annotations for text with various orientations. (Example: Fig.2.5b)
- **SynthText**: A synthetic dataset generating large-scale annotated data by overlaying text on natural scene images. (Example: Fig.2.5a)

For Indic texts, datasets like those for Devanagari or Tamil are scarce, often requiring synthetic data generation or transfer learning from general datasets [19].

(a) SynthText Dataset

(b) TotalText Dataset

Figure 2.5: Sample Images from the datasets

Table 2.1: Summary of Datasets for Scene Text Detection

Dataset	**Description**	**Key Features**
ICDAR Series	Benchmarks for text localization and recognition	Multi-lingual, curved text
COCO-Text	Large-scale natural images	Diverse scenarios
Total-Text	Curved text focus	Irregular text shapes
CTW-1500	Curved text dataset	Natural scenes
SynthText	Synthetic data	Large-scale annotations

2.2.4 Challenges

Below is a list of common challenges faced in STD, with a focus on their implications for detecting texts in natural images:

- **Variability in Text Appearance:** Scene text comes in different variations like font, size, color, and style, which complicates detection process. These variations challenge models to generalize across diverse text appearances [23].
- **Complex Backgrounds and Clutter:** Natural scenes frequently contain complex background which obscure textual content. Sometimes, the background clutter can mimic text-like patterns, leading to false-positives [33].
- **Arbitrary Text Shapes and Orientations:** Text may come in different forms or perpectives in natural environment. It can appear in forms such as curved, rotated, or distorted. Conventional bounding box approach struggle with such irregularities. [20].

- **Low Resolution and Blur:** Images always have a chance of containing low resolution and blurry images. It can sometimes by due to motion or unable to focus properly. This degrades the text legibility [33].
- **Multilingual and Multiscript Scenarios:** Scenes may contain text in multiple languages or scripts, increasing the complexity of detection. In Indic contexts, mixed-script scenarios (e.g., Hindi with English) are common, requiring models to differentiate between scripts with overlapping visual features [23].
- **Limited Annotated Data for Indic Scripts:** Deep learning models require large annotated datasets for training, but such datasets are scarce for Indic scripts compared to Latin-based scripts. The lack of diverse, high-quality annotations hinders the development of robust STD models for languages like Bengali or Telugu [23].

2.2.5 Evaluation Metrics

Evaluation metrics for scene text detection assess the accuracy of detected text regions compared to ground truth:

- **Precision** is defined as the ratio of correctly detected regions to the total number of detected regions, expressed as:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}.$$

- **Recall** measures the ratio of correctly detected regions to the total number of ground truth regions, given by:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}.$$

- **F-Measure** is the harmonic mean of precision and recall, providing a balanced measure of detection accuracy [33]. It is calculated as:

$$\text{F-Measure} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}.$$

2.3 Scene Text Recognition

2.3.1 History of Early Methods

Early scene text recognition methods relied on hand-crafted features, such as histogram of oriented gradients (HOG), connected components (see Fig.2.2), and stroke width transform (see Fig.2.3). These approaches were limited by their inability to handle the variability of natural scenes, including diverse fonts, orientations, and complex backgrounds, resulting in lower accuracy compared to modern methods [5].

2.3.2 Recent Advancements

Scene Text Recognition (STR) has witnessed significant progress with the advent of deep learning, particularly through Convolutional Neural Networks (CNNs) and Transformer-based architectures.

2.3.2.1 CNN-Based Approaches

CNNs have been instrumental in STR, primarily due to their prowess in feature extraction. Recent enhancements include:

- **CRNN Enhancements**: Recent improvements integrate spatial transformers and Inception ResNet for robust feature extraction, showing improved performance on IndicSTR12 for regular Indic texts but struggling with irregular layouts [2].
- **CRAFT**: Adapted for end-to-end STR, CRAFT's character region prediction excels in detecting multi-oriented and multilingual text, suitable for Indic scripts with complex layouts [1].
- **RobustScanner**: Incorporates position-enhanced attention for irregular text, promising for Indic scenes but requires fine-tuning on script-specific datasets [34].
- **SPIN**: Preserves text structure for curved text recognition, effective for Indic multi-oriented text scenarios [32].

2.3.2.2 Transformer-Based Approaches

Transformers have revolutionized STR by capturing long-range dependencies and contextual information. Notable developments include:

- **TrOCR**: Uses ViT encoder and RoBERTa decoder for end-to-end recognition, achieving state-of-the-art results on complex layouts after fine-tuning on SCUT-CTW1500. Promising for Indic STR with multilingual adaptability [14].
- **Parseq**: Employs ViT-based encoder-decoder with Permutation Language Modeling, outperforming CRNN on IndicSTR12 across 12 Indian languages due to robust multi-head attention [2] [21].
- **SVTR**: Single ViT model with CTC loss and enhanced attention, offering high accuracy and speed for multi-oriented text [8].
- **VIPTR**: ViT-based with Manhattan self-attention, efficient for cross-language STR [4].
- **LSTR**: Lightweight DeiT-tiny model with position-enhancement, offering 3% higher accuracy than baselines[16].

2.3.2.3 Hybrid and Emerging Approaches

The fusion of CNNs and Transformers has led to hybrid models that capitalize on the strengths of both architectures:

- **SwinTextSpotter**: Integrates Swin Transformer-based detection with recognition modules, achieving superior performance in end-to-end text spotting tasks [10].
- **ESTextSpotter**: Employs explicit synergy between detection and recognition tasks within a Transformer framework, enhancing overall STR performance [11].

Some studies categorize STR approaches as segmentation-based (context-free, focusing on individual character recognition) or sequence-based (context-aware, leveraging contextual and linguistic dependencies for holistic text decoding) [2].

- **Context-Free Scene Text Recognition (STR):** This approaches recognizes the text in images only based on visual features of individual characters or words. It has no knowledge on language semantics when predicting the caracters. It relies on models like CNN or Connectionist Temporal Classification to decode text. It works ideally in isolated text instances.

- **Context-Aware Scene Text Recognition (STR):** This approach recognizes the text in images with visual features and having a prior knowledge of language semantics. Language models use external knowledge to refine predictions, improving performance in challenging scenarios involving noise or irregular texts.

For Indic texts, models are being tailored to handle complex character conjuncts and script diversity, often using transfer learning due to limited annotated data [5].

2.3.3 Datasets

The following datasets are widely used for scene text recognition tasks, with specific relevance to Indic scripts where applicable.

- **SynthText**: A synthetic dataset with over 800,000 images containing text overlaid on natural scenes. It includes diverse fonts, sizes, and orientations, making it suitable for pre-training deep learning models for text detection and recognition. While not specific to Indic scripts, it is valuable for general scene text recognition tasks (See fig.2.5a) [9].

- **ICDAR Datasets**: The ICDAR series includes datasets like ICDAR 2003, 2013, 2015, and 2019 (ArT and MLT). ICDAR 2015 contains 1,000 training and 500 testing images with text in challenging conditions. ICDAR 2019 MLT supports multiple scripts, including Indic scripts like Devanagari and Tamil, with 18,000 images [12, 26].

- **IndicSTR12**: A dataset for scene text recognition in 12 Indian languages, including Hindi, Tamil, Telugu, and Bengali. It features real-world images with text on signboards and billboards, ideal for evaluating models on Indic scripts (See Fig.2.6) [21].

Figure 2.6: IndicSTR12 Dataset Example

- **IIIT-5K**: Contains 5,000 images (2,000 training, 3,000 testing) from natural scenes, primarily with English text but including some Indian language text. It provides word-level annotations for text detection and recognition [24].
- **COCO-Text**: Built on MS COCO, it includes over 63,000 images with 173,000 text instances. It is designed for scene text detection and recognition in natural images and can be fine-tuned for Indic scripts [30].
- **Total-Text**: Contains 1,555 images with over 11,000 text instances, including curved and multi-oriented text, suitable for complex text recognition tasks [6].

Table 2.2: Summary of Datasets for Scene Text Recognition

Dataset	Description	Image Count	Indic Support
SynthText	Synthetic images with overlaid text	800,000+	No
ICDAR 2015	Real-world images with challenging conditions	1,500	Partial (MLT)
IndicSTR12	Indic script text in real-world scenes	27,000	Yes
IIIT-5K	Natural scenes with some Indian text	5,000	Partial
COCO-Text	Natural images with text annotations	63,000+	No
Total-Text	Curved and multi-oriented text	1,555	No

2.3.4 Challenges

Similar to challenges faced for detection models, similar are applicable for recognition models as they are both dependent of quality of image and textual content. Following are some common challenges faced when developing Recognition models:

- **Diverse Visual Styles and Fonts:** Scene texts comes in various forms, like, fonts, colors, size, etc. This makes it hard for models to generalize. Considering the scripts of Indian Languages makes the scenario much complicated. [33]
- **Image Noise and Degradation:** Poor Image and Textual content quality lead to poor predictions. They are often suffered from blur, low resolution and sometimes ligting issues. Considering the Indian Scripts even slightest noise can change the whole characters which leads to significat prediction errors. [33]
- **Irregular Text Layouts and Orientations:** Scene Text in curved, tilted or disstorted form complicate recognition process. As they need extra preprocessing to get the rich features before used for recognition. [33]
- **Multilingual and Mixed-Script Environments:** Many Indian scenes feature multiple languages side by side — think of a shop sign written in both Hindi and English. This creates confusion for recognition models, especially when characters from different scripts share visual similarities [23].
- **Lack of Richly Annotated Indic Datasets:** Training robust deep learning models requires large, diverse datasets with accurate labels. Unfortunately, such datasets are limited for many Indic languages. [23].
- **Linguistic and Morphological Complexity:** Indic languages are morphologically rich and often context-dependent. Features like compound characters, inflections, and complex word boundaries require the model to learn deeper, long-range patterns in order to make accurate predictions [33].

2.3.5 Evaluation Metrics

Several metrics are employed to quantify the effectiveness of STR systems, with Character Error Rate (CER) and Word Error Rate (WER) being the most prominent due to their ability to measure performance at character and word levels, respectively.

- **Character Error Rate (CER):** The Character Error Rate (CER) evaluates the accuracy of a text recognition system at the character level. It measures the number of edit operations (insertions, deletions, and substitutions) needed to align the predicted text with the ground truth, normalized by the ground truth length. The CER is defined as:

$$\text{CER} = \frac{S + D + I}{N}, \tag{2.1}$$

  where S is the number of substitutions, D is the number of deletions, I is the number of insertions, and N is the total number of characters in the ground truth. For Indic scripts, which feature complex character sets, conjunct consonants, and diacritics, CER is critical for assessing character-level accuracy, directly impacting readability [23].

- **Word Error Rate (WER):** The Word Error Rate (WER) assesses performance at the word level, using the Levenshtein distance with words as the basic unit. It is defined as:

$$\text{WER} = \frac{S_w + D_w + I_w}{N_w}, \tag{2.2}$$

  where S_w, D_w, and I_w represent the number of word substitutions, deletions, and insertions, respectively, and N_w is the total number of words in the ground truth. WER is vital for evaluating the usability of recognized text in applications like text translation, especially for Indic languages with morphological complexity and script-specific word formations [23].

- **Other Metrics:** Beyond CER and WER, metrics like Normalized Edit Distance (NED) and Total Edit Distance (TED) are occasionally used. NED normalizes the edit distance by the length of the longer string, while TED reports the raw edit distance. Additionally, Correctly Recognized Words (CRW) and Accuracy measure the percentage of perfectly recognized words or characters, providing intuitive performance insights [12].

Chapter 3

METHODOLOGY AND IMPLEMENTATION

This section outlines the step-by-step approach adopted for developing and evaluating the scene text detection and recognition system. It includes a description of the system configuration, the hardware and software tools used, and a breakdown of the specific methods applied for both detection and recognition tasks.

3.1 System Requirements

The specifications of the workstation used to execute the model are illustrated by table 3.1:

Table 3.1: System Specifications

Processor	:	Intel® Core™ i9-14900K 32 cores @ 3.20GHz
System RAM	:	128 Gigabytes of DDR4
Graphic Processing Unit	:	NVIDIA® A6000 + Intel® UHD Graphics 770
Video RAM of the GPU	:	49 Gigabytes (48 GB GDDR6 + 1 GB shared)
Operating System	:	Microsoft™ Windows 11 Pro for Workstations
CUDA version	:	NVIDIA® CUDA 12.4
Python version	:	Python 3.12.9

These tools provide the necessary functionality for data processing, model training, and evaluation.

Table 3.2: Important Python Packages for Scene Text Extraction

Package	**Version**
torch	2.6.0+cu126
torchvision	0.21.0+cu126
transformers	4.49.0
numpy	2.1.2
opencv-python	4.11.0.86
pandas	2.2.3
scikit-learn	1.6.1
scipy	1.15.2
matplotlib	3.10.1
indic-nlp-library	0.92
sentencepiece	0.2.0
huggingface-hub	0.29.1
safetensors	0.5.3
tqdm	4.67.1

3.2 Detection

3.2.1 Model Architecture

For Scene Text Detection, we incorporate DBNet[15] Methodology on UperNet Architecture[31]. We further train on two different backbones which are used in UperNet namely, ConvNext [18] and Swin Transformer [17]. UperNet is a unified perceptual parsing framework designed to segment diverse visual concepts, is proposed as instance segmentation of objects in a complex scenes. We adapt the same by considering the objects as text. DBNet employs a differentiable binarization module to accurately predict text regions. It allows the model to learn optimal thresholds during training (See Eq.3.1). The integrated model architecture incorporates feature extractions, feature pyramid network for handling multi-scale text instances, and segmentation head.

To generate the final text detection outputs, the model produces probability and threshold maps. The binary map is extracted by computing the sigmoid difference between these maps, as described in [15]:

$$B = Sigmoid(P - T) \tag{3.1}$$

where B is the binary map, P is the probability map, T is the threshold map, and σ denotes the sigmoid function. This process highlights text regions with high confidence.

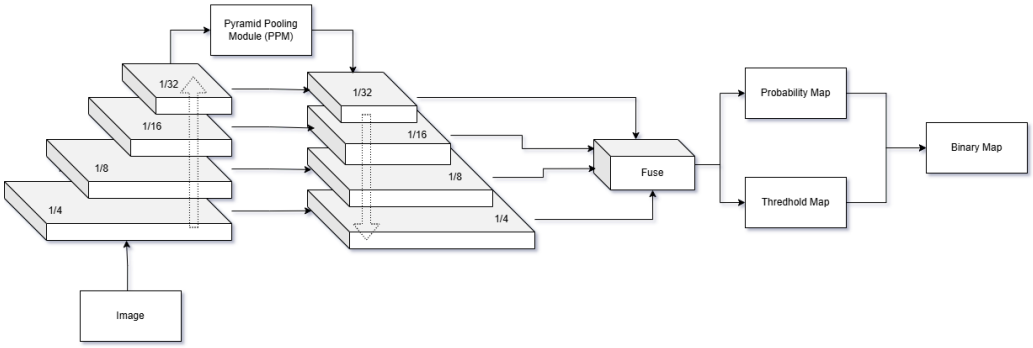


Figure 3.1: Proposed Model

We experiment with two different backbones:

1. **ConvNext [18]:** A modern convolutional neural network that incorporates design choices from vision transformers, achieving state-of-the-art performance on image classification tasks.
2. **Swin Transformer [17]:** A hierarchical vision transformer that uses shifted windows to capture both local and global dependencies efficiently.

By integrating these powerful backbones, we aim to enhance the feature extraction capabilities of our proposed model, leading to improved detection performance, particularly in complex scenes with varied text sizes and orientations.

Datasets:

The models are trained and evaluated on the following datasets:

- **SynthText [9]:** A large-scale synthetic dataset containing images with overlaid text, used for pre-training.
- **TotalText [6]:** A dataset featuring curved and oriented text instances, essential for evaluating the detection of arbitrarily shaped text.
- **IndicSTR12 [21]:** A comprehensive dataset for Indic scene text recognition, covering 12 major Indian languages, used to assess multilingual text detection performance.

3.2.2 Preprocessing

Accurate ground truth annotations, especially segmentation masks, are essential for training robust scene text detection models. However, not all datasets come with these masks readily available. For datasets such as SynthText [9] and IndicSTR12 [21], which only offer bounding box annotations, we generate rectangular masks to approximate the text regions. While this simplification speeds up preprocessing and allows training to proceed, it does come with a trade-off: the model may struggle to accurately detect text that is curved, skewed, or irregular in shape.

To compensate for this, we fine-tuned our model using the TotalText dataset [6], which includes high-quality segmentation masks for English texts that are curved or multi-oriented. This helps the model learn more nuanced text boundaries in real-world scenarios.

Unfortunately, for Indic scripts, such richly annotated datasets are still not available. As a result, we had to rely on rectangular masks for IndicSTR12, which limits the model's ability to generalize well to the complex structures found in Indic texts. Addressing this gap either by creating synthetic segmentation data or investing in manual annotation presents a valuable direction for future work.

3.2.3 Hyperparameter Configuration

The training process is governed by carefully selected hyperparameters to ensure stable convergence and optimal performance. The configurations are as follows:

- **Batch Size**: 8, balancing computational efficiency and gradient stability.
- **Learning Rate**: 1×10^{-4}, providing a moderate pace for parameter updates.
- **Optimizer**: AdamW, an adaptive optimizer with weight decay to prevent overfitting.
- **Loss Function**: DBLoss, specifically designed for scene text detection tasks, as detailed below.

3.2.4 Loss Function: DBLoss

The Differentiable Binarization (DB) Loss, proposed in DBNet [15], is employed to optimize the model for text detection. DBLoss integrates three components: a probability map loss, a binary map loss, and a threshold map loss, enabling precise text boundary delineation. The total loss is defined as:

$$L = L_{\text{prob}} + \alpha L_{\text{bin}} + \beta L_{\text{thr}} \tag{3.2}$$

where:

- L_{prob} is the loss for the probability map, computed using binary cross-entropy to predict text regions.
- L_{bin} is the loss for the binary map, also using binary cross-entropy, applied after differentiable binarization to refine text boundaries.
- L_{thr} is the loss for the threshold map, calculated using L1 distance to ensure accurate boundary thresholds.
- $\alpha = 1.0$ and $\beta = 10.0$ are weighting factors to balance the contributions of each loss component.

This formulation ensures robust text detection by optimizing both the segmentation and binarization processes, critical for handling the intricate structures of Indic scripts.

3.2.5 Implementation

The foundation of the implementation is the UPerNet architecture, known for its flexibility and strong performance in general scene segmentation tasks [31]. we use a pre-trained UPerNet model available through the Hugging Face Transformers library, and integrated it into the project using the PyTorch framework. We then extend the pretrained UperNet model with DBNet [15] head, which predicts the probability and threhold maps. We use the generated two maps to predict the binary map (see Eq.3.1). This highlights the regions with high precision.

3.2.6 Training and Evaluation

The training strategy followed a two-stage approach. First, the model is pretrained on the large-scale synthetic SynthText dataset, which helped it learn

general features of scene text. Once the model had a solid foundation, it is fine-tuned on TotalText and IndicSTR12 datasets. This allowed the model to adapt to more realistic conditions and, more importantly, to learn the nuances of Indic scripts and multilingual settings.

For evaluation, we use standard text detection metrics — Precision, Recall, and F1-Score — to measure how well the model could detect text across various conditions and scripts. These metrics helped in assessing both the accuracy and robustness of the model in handling real-world scene images.

3.3 Recognition

3.3.1 Model Architecture

This study adopts the TrOCR (Transformer-based Optical Character Recognition) framework (See Fig.3.2), which integrates a vision encoder with a text decoder for end-to-end scene text extraction, specifically tailored for Indic texts [14]. We explore two vision encoders: the Data-Efficient Image Transformer (DeiT) and the Swin Transformer, each paired with the IndicBART decoder.

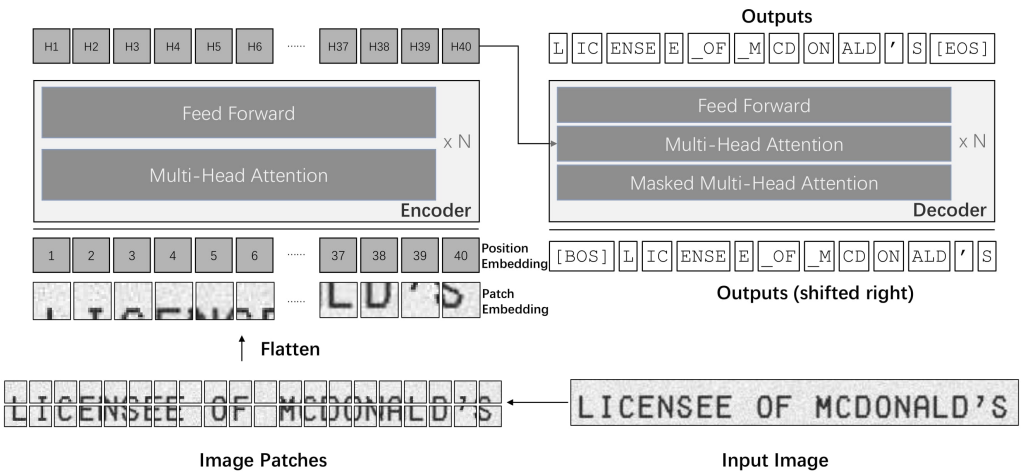


Figure 3.2: TrOCR Architecture

- **Vision Encoders:**
  - **DeiT:** A Vision Transformer model that employs a teacher-student framework with a distillation token for efficient training [29]. It pro-

cesses input images by dividing them into 16x16 patches, applying transformer layers to capture global dependencies.

  - **Swin Transformer:** A hierarchical Vision Transformer that uses shifted windows for local self-attention, enabling efficient computation for high-resolution images [17]. It splits images into 4x4 patches, constructing a multi-scale feature representation.

- **Text Decoder:**

  - **IndicBART:** A sequence-to-sequence model pre-trained on 11 Indic languages and English, based on the mBART architecture [7]. Its specialized training on Indic corpora makes it adept at generating text in Indic scripts.

The choice of IndicBART was informed by experiments with other multilingual models, such as XLMRoberta and mBART, which demonstrated inferior performance in capturing the linguistic nuances of Indic scripts. To handle multiple scripts, we utilize special tokens inherent to the IndicBART tokenizer, such as `<2as/>` for Assamese and `<2te/>` for Telugu. The output sequence is structured as

```
[Script Token] [Text Tokens] [EOS Token]        (3.3)
```

enabling the model to generate both the script identifier and the corresponding text. For example,

```
<2te> ఉరిమి గెలుపు! </s>                       (3.4)
```

tells us that is is a Telugu Scipt and it reads ఉరిమి గెలుపు! (translation, ”Roar to Victory”).

Due to limitations of IndicBART, which is primarily optimized for Indic Languages, non-Devanagari Script texts are transliterated into Devanagari script prior to training [7] (See Fig.3.3). This preprocessing step ensures compatibility with the model's tokenizer and leverages its robust performance on Devanagari-based inputs. The model is trained on these transliterated Devanagari texts. During inference, the generated Devanagari text is converted back to the target Indic script using the corresponding language token. Instead of having another model to identify languages, we are using language token to identify the language allowing is to reverse the transliteration process.

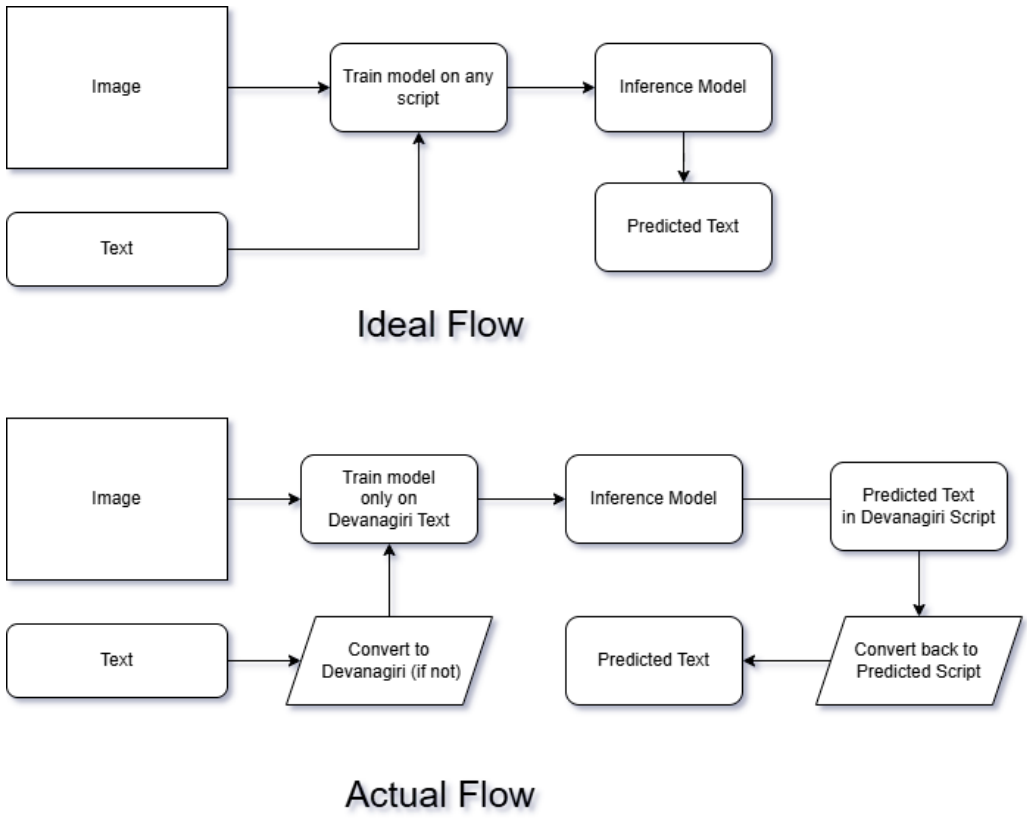


Figure 3.3: Ideal vs. Actual Workflow when Training Recognition Model due to the Limitation of IndicBART

3.3.2 Preprocessing

The preprocessing pipeline prepares both image and text data to meet the requirements of the vision encoders and text decoder.

- **Image Preprocessing:**
  - Images are resized to a uniform resolution of 224x224 pixels to ensure consistency.
  - Normalization is applied using the mean and standard deviation values from the ImageNet dataset to standardize pixel values.
- **Text Preprocessing:**
  - Ground truth text is first converted if it is a non-Devanagiri Indic Text to Devanagiri Text.
  - Ground truth text is tokenized using the IndicBART tokenizer, which includes special tokens for script identification.

– The target sequence for training is formatted as [Script Token] followed by text tokens and terminated with the [EOS] token (See Eq.3.3).

3.3.3 Hyperparameters Configuration

The training process employs the following hyperparameters:

- **Batch Size:** 8
- **Optimizer:** AdamW with a learning rate of 5×10^{-5} and weight decay of 0.01
- **Decoder Generation Parameters:**
  - Maximum sequence length: 256
  - Early stopping: Enabled
  - No repeat n-gram size: 3
  - Length penalty: 2.0
  - Number of beams: 4
- **Loss Function:** The total loss function for the text recognition module is designed to optimize token prediction for Indic scripts, combining a weighted L1 loss for language-specific token prediction with a cross-entropy loss for classification. The improved loss function is defined as:

$$L_{\text{total}} = \alpha L_{\text{L1}}^{\text{token}}(\hat{y}, y) + L_{\text{CE}}(\hat{p}, p) \tag{3.5}$$

  where:
  - $L_{\text{L1}}^{\text{token}}(\hat{y}, y) = |\hat{y}_0 - y_0|$ is the L1 loss, measuring the absolute difference between the predicted token $\hat{y}_0$ and the ground truth token y_0.
  - $L_{\text{CE}}(\hat{p}, p) = -\sum_{c=1}^{C} p_c \log(\hat{p}_c)$ is the cross-entropy loss, comparing the predicted probability distribution $\hat{p}$ over C classes (e.g., characters or tokens) with the ground truth distribution p.
  - α is a weighting factor (set to 10.0 in this implementation) to balance the contributions of the L1 and cross-entropy losses.
  - $\hat{y}$ and y denote the predicted and ground truth token embeddings, respectively, capturing language-specific features of Indic scripts.
  - $\hat{p}$ and p represent the predicted and ground truth probability distributions over the token classes, respectively.

3.3.4 Implementation

The implementation leverages pre-trained models integrated into an encoder-decoder framework.

- **Model Initialization:** Pre-trained weights for DeiT [29], Swin Transformer [17], and IndicBART [7] are loaded from their respective repositories.
- **Encoder-Decoder Architecture:** The vision encoder's output is fed into the IndicBART decoder to generate text sequences autoregressively.
- **Custom Loss Function:** A modified cross-entropy loss is implemented, where the loss for the script token is multiplied by a factor of 10 to enhance its influence.
- **Checkpointing:** Due to the extensive training duration, model checkpoints are saved periodically between batches to preserve progress.

3.3.5 Training and Evaluation

Firstly, the model is trained on DeIT + IndicBART model. Using transformers library to simplify the process by using `VisionEncoderDecoder` class. Appropriate tokenizer and preprocessor are obtained similarily using `AutoTokenizer and AutoProcessor`. The model is then loaded with their respective pretrained weights. We then finetune with model with Input Image and Output Text. The text is tokenized as part of training.

We do this process similarily for the other combination of the model which is Swin + IndicBART.

Evaluation: Performance is assessed using standard OCR metrics, including Character Error Rate (CER) and Word Error Rate (WER), to evaluate text recognition accuracy across various Indic scripts.

Chapter 4

RESULTS AND DISCUSSIONS

This section outlines the outcomes of the scene text detection and recognition experiments conducted using deep learning models customized for handling Indic scripts. The goal was to assess how well these models perform in realistic, multilingual environments.

For the detection task, models are evaluated using two popular vision backbones — ConvNext and Swin Transformer — and compared their performance to a baseline DBNet model. The models were trained and fine-tuned across a mix of datasets, including SynthText, TotalText, and IndicSTR12, to ensure both robustness and relevance to Indic scripts.

In the recognition task, different vision encoders were tested within the TrOCR framework to understand how effectively they could transcribe text from the detected regions. The experiments aimed to uncover not just accuracy but also real-world usability — especially for low-resource scripts and noisy image conditions. The following sections break down these findings and provide commentary on what worked well, what didn't, and why.

4.1 Scene Text Detection

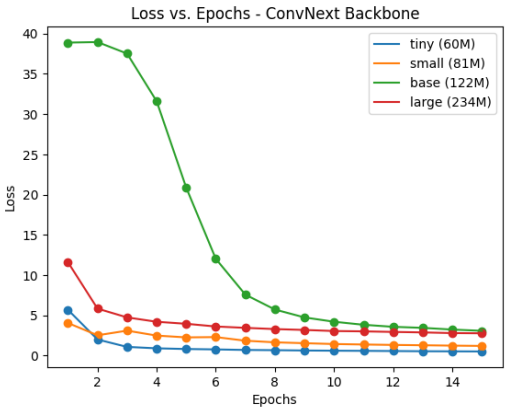
This part of the study focuses on how well the proposed model can localize text in natural scene images, particularly those featuring Indic scripts. Using ConvNext and Swin Transformer backbones, the model was put to the test against a strong baseline — DBNet — to see if the integration of transformer-based features offered meaningful improvements in complex scenarios.

4.1.1 Results on TotalText Dataset

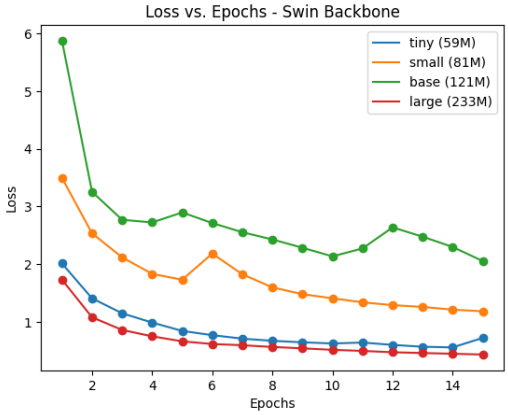
The performance of different model variants trained exclusively on the TotalText dataset is summarized in Table 4.1. The table includes metrics such as precision, recall, and F1-score, which provide insights into the model's detection capabilities. Figure 4.1 illustrates sample detection outputs, showcasing the model's ability to generate text masks in various scenarios.

Table 4.1: Performance metrics for Swin and ConvNext backbone variants trained on the TotalText dataset.

Metric	Swin Backbone				ConvNext Backbone			
	Tiny	**Small**	**Base**	**Large**	**Tiny**	**Small**	**Base**	**Large**
Precision	0.8359	0.8207	0.7594	0.8824	0.8588	0.8202	0.6284	0.8809
Recall	0.7616	0.7555	0.4969	0.8259	0.8166	0.7419	0.6166	0.1869
F-Score	0.7970	0.7868	0.6008	0.8532	0.8372	0.7791	0.6224	0.3084



(a) ConvNext Loss vs. Epoch



(b) Swin Loss vs. Epoch

Figure 4.1: Graphs for both the backbones Loss vs. Epoch

4.1.2 Results on SynthText Dataset

Training the model on the SynthText dataset yields the performance metrics presented in Table 4.2. Figure 4.2 provides the loss comparison between the *tiny* and *base* variants of the Swin Backbone-based model.

The SynthText-trained models exhibit superior performance compared to those trained on TotalText, likely due to the larger and more diverse synthetic

Table 4.2: Performance metrics for Tiny and Base variants trained on the TotalText dataset.

Metric	Tiny	Base
Precision	0.8352	0.8628
Recall	0.8411	0.8521
F-Score	0.8381	0.8574

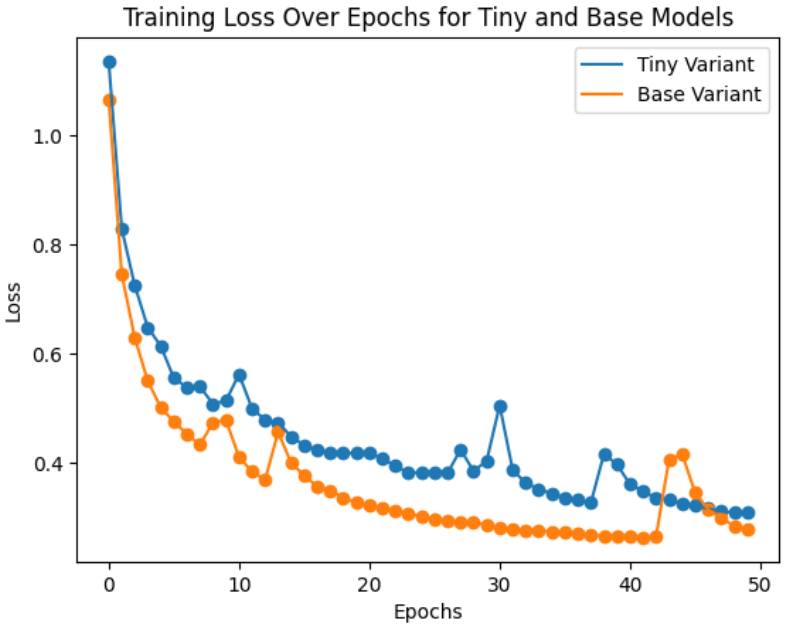


Figure 4.2: Loss Comparision between tiny and base variants of Swin Backbone based model

dataset. The Swin backbone base model demonstrates the best performance, with an F1-score of 0.85.

4.1.3 Fine-Tuning with TotalText and IndicSTR12

Further experiments involve training the model on SynthText and fine-tuning it with TotalText and IndicSTR12 datasets. The results, illustrated in Figure 4.3, demonstrate the model's ability to handle complex scenarios involving Indic texts.

A real-world test case, captured using a mobile phone, is shown in Figure 4.4a. The model successfully detects text in this unconstrained environment, highlighting its practical applicability. Additionally, Figure 4.4b showcases the detection of Indic text, including scripts such as Devanagari and Tamil.

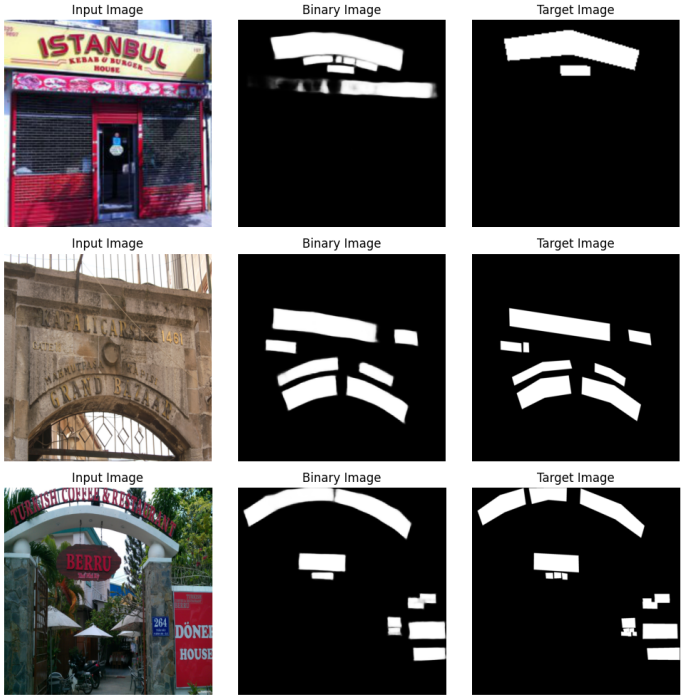
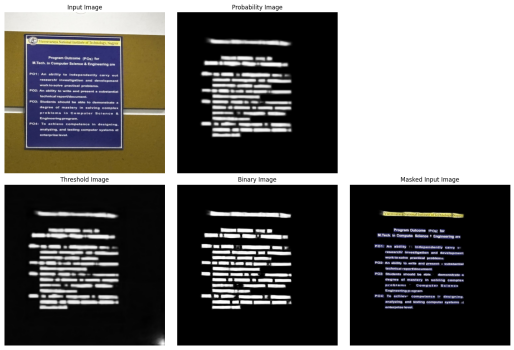
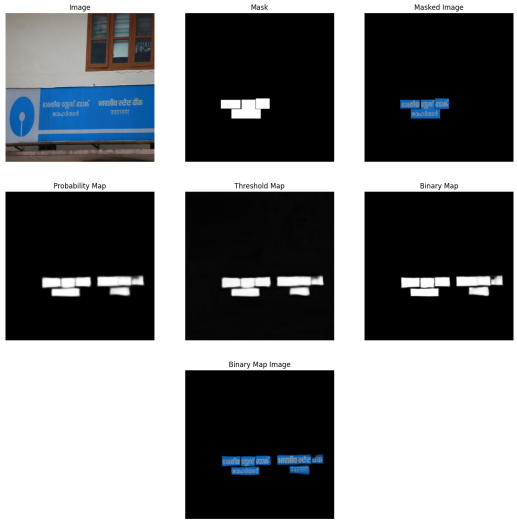


Figure 4.3: Different Examples compared to their ground truths



(a) Example of image captured using Mobile Phone



(b) Example of Detection of Indic Text

Figure 4.4: More Examples of Scene Text Detection

4.1.4 Discussion

The detection results reveal that the proposed model, with ConvNext and Swin backbones, significantly outperforms the DBNet baseline across all datasets. A notable observation is the model's ability to detect text masks that were

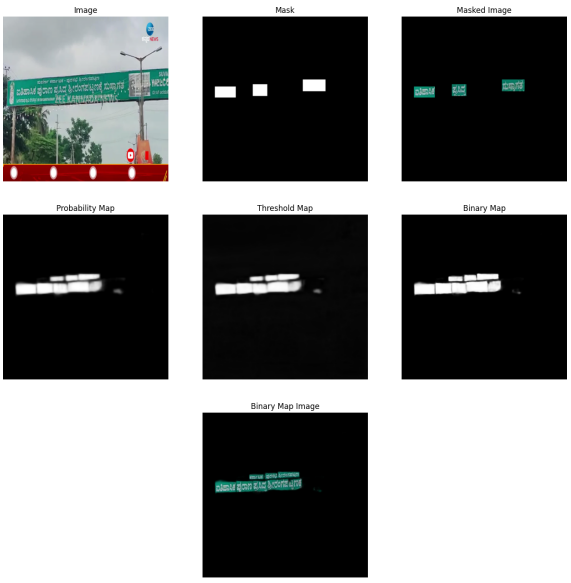


Figure 4.5: Example of Inconsistent Annotations

not annotated in the datasets (See Fig.4.5), particularly in images containing multiple languages. For instance, in the IndicSTR12 dataset, annotations are provided only for a specific language, yet the model identifies texts in other languages present in the same image. This robustness suggests that the model learns generalized text features, likely due to the diverse training data and transformer-based architectures.

As shown in Table 4.1, the Swin Transformer backbone consistently outperformed its ConvNext counterpart across most model sizes when trained on the TotalText dataset. Among all configurations, the Swin Large variant achieved the best results, with a precision of 0.8824, recall of 0.8259, and an F-Score of 0.8532. These metrics indicate strong capability in detecting complex, multi-oriented text, particularly when sufficient context is provided through Swin's hierarchical attention mechanisms.

On the other hand, the ConvNext Large variant showed a noticeable drop in performance, with an F-Score of just 0.3084 — largely due to a low recall of 0.1869. This suggests that the model struggled to capture all relevant text regions, especially in images with dense or irregular text placements. The gap in performance between the two architectures highlights how transformer-based models may offer a stronger inductive bias for capturing text patterns in real-world scenes.

After pretraining on the synthetic SynthText dataset and fine-tuning on TotalText and IndicSTR12, the results in Table 4.2 reveal a more stable perfor-

mance across variants. The Swin Base model achieved the highest F-Score of 0.8574, while the Tiny variant was close behind at 0.8381. While neither variant reached absolute state-of-the-art levels, both significantly outperformed the DBNet baseline, confirming that the proposed architecture and training pipeline provide a meaningful improvement, particularly for Indic script scenarios.

Despite these improvements, certain edge cases remain challenging. For example, when texts in multiple languages appear close together within a scene, the model occasionally detects only one script — likely due to overlapping character features or limited multilingual exposure during training. However, one encouraging observation is the model's strong performance in detecting smaller text regions (see Fig. 4.6), an area where the baseline models often fall short.

These results emphasize the critical role of diverse and well-annotated datasets, especially for underrepresented Indic scripts. They also suggest that transformer-based backbones, when fine-tuned properly, hold considerable promise for improving scene text detection in multilingual and complex visual environments.

Figure 4.6: Example of detecting small texts

4.2 Scene Text Recognition

The recognition experiments leverage the TrOCR framework, which employs vision transformer encoders to transcribe detected text regions into readable

text. The performance is evaluated across various Indic languages, with a focus on scripts such as Devanagari, Tamil, and Telugu.

4.2.1 Results

The recognition performance of different vision transformer encoders is summarized in Table 4.3 and Table 4.4, which report metrics such as character error rate (CER) and word error rate (WER).

Language	CER	WER	CRR	WRR
Assamese	0.069	0.205	0.931	0.795
Bengali	0.133	0.395	0.867	0.605
Gujarati	0.058	0.192	0.942	0.808
Hindi	0.075	0.283	0.925	0.717
Kannada	0.212	0.576	0.788	0.424
Malayalam	0.154	0.519	0.846	0.481
Marathi	0.082	0.312	0.918	0.688
Odia	0.120	0.375	0.880	0.625
Punjabi	0.097	0.304	0.903	0.696
Tamil	0.122	0.409	0.878	0.591
Telugu	0.220	0.612	0.780	0.388

Table 4.3: CER and WER for DeIT Encoder for TrOCR

Language	CER	WER	CRR	WRR
Assamese	0.074	0.206	0.926	0.794
Bengali	0.140	0.398	0.860	0.602
Gujarati	0.062	0.187	0.938	0.813
Hindi	0.080	0.297	0.920	0.703
Kannada	0.219	0.570	0.781	0.430
Malayalam	0.138	0.448	0.862	0.552
Marathi	0.083	0.298	0.917	0.702
Odia	0.137	0.378	0.863	0.622
Punjabi	0.105	0.314	0.895	0.686
Tamil	0.142	0.434	0.858	0.566
Telugu	0.242	0.634	0.758	0.366

Table 4.4: CER and WER for Swin Encoder for TrOCR

The Character Recognition Rate (CRR) and Word Recognition Rate (WRR) measure the accuracy of a model, akin to precision. They are calculated as $CRR = 1 - CER$ and $WRR = 1 - WER$, where CER is the Character Error Rate and WER is the Word Error Rate. Higher CRR and WRR values indicate better model performance. Recognition outputs are classified into three categories: good, moderate, and poor predictions.

4.2.1.1 Good Examples

	Text	Language
Ground Truth	मंदीर	Hindi
Prediction	मंदीर	Hindi
Translated	मंदीर	

	Text	Language
Ground Truth	माटुंगा	Marathi
Prediction	माटुंगा	Hindi
Translated	माटुंगा	

	Text	Language
Ground Truth	নামানে	Assamese
Prediction	নামানে	Assamese
Translated	নামানে	

	Text	Language
Ground Truth	ಒರಾಯನ್	Kannada
Prediction	ओरायन्	Kannada
Translated	ಒರಾಯನ್	

4.2.1.2 Moderate Examples

	Text	Language
Ground Truth	తిరుపతి	Telugu
Prediction	तिरुवति	Telugu
Translated	తిరువతి	
Difference	తిరుపవతి	

	Text	Language
Ground Truth	മാവേലിക്കര	Malayalam
Prediction	माळेलिक्कर	Malayalam
Translated	മാളേലിക്കര	
Difference	മാവളേലിക്കര	

	Text	Language
Ground Truth	ସଫାରୀ	Odia
Prediction	सफीर	Odia
Translated	ସଫାର	
Difference	ସଫ ◌ା ◌1ର ◌1	

	Text	Language
Ground Truth	ধিক্কার	Bengali
Prediction	ধিক্কা□	Assamese
Translated	ধিক্কাৰ	
Difference	ধিক্কারৰ	

	Text	Language
Ground Truth	கோபாலகிருஷ்ணன்	Tamil
Prediction	कोपाल्पाकुकिरुष्ञरमणम्	Tamil
Translated	கோபால்பாகுகிருஷ்ஞரமணம்	
Difference	கோபால்பாகுகிருஷ்ஞர்மணனம	

4.2.1.3 Poor Examples

	Text	Language
Ground Truth	મહેસાણા	Gujarati
Prediction	मुहंसाया	Marathi
Translated	मुहंसाया	

	Text	Language
Ground Truth	ਘੁੰਨਮ	Punjabi
Prediction	घुंनमन	Hindi
Translated	घुंनमन	

	Text	Language
Ground Truth	ରାୟଗଡ଼ା	Odia
Prediction	ଷଣ୍ଢଗଡ଼ି	Odia
Translated	ଷଣ୍ଢଗଡ଼ି	
Difference	ର◌ାୟଷଣ◌ଢଗଡ଼◌ା◌ି	

4.2.2 Discussion

The recognition results, particularly for South Indian languages like Tamil and Telugu, were noticeably lower than expected. While this outcome isn't entirely surprising given the complexity of these scripts it brings to light some critical limitations in the current pipeline. One likely reason for the underperformance is the unintended transliteration of certain texts into the Devanagari script during preprocessing. This not only introduces inconsistencies but also alters the visual and linguistic structure of the original scripts, making it harder for the model to learn accurate character representations.

This issue highlights the importance of implementing script-specific preprocessing pipelines that can preserve the unique orthographic features of each language. South Indian scripts, for example, rely on compound characters, intricate diacritics, and spatial arrangements that differ significantly from scripts like Devanagari. A one-size-fits-all approach in preprocessing tends to dilute these distinctions and, as shown by the results, can negatively affect recognition accuracy.

Another notable challenge was the confusion between visually or linguistically similar scripts for instance, Hindi and Marathi, which both use Devanagari. A different example would be Assamese and Bengali; both share a similar script called Easter Nagri. The model occasionally misidentifies text when characters or words overlap visually, even though the languages themselves are distinct. This suggests that the current TrOCR framework, while generally effective for script recognition, struggles with nuances that go beyond character shape alone.

We also see that although the CRR is high but the WRR is comparitively lower. This indicates that the model is making mistakes when predicting the entire word by making wrong character predictions. This suggests that more training and further improvements in the model architecture and/or dataset is required to improve the robustness.

These findings underscore the need for more refined strategies, such as language-specific fine-tuning and targeted data augmentation. Incorporating more diverse training samples for underrepresented scripts and enhancing the quality of script labels could also help reduce error rates. Overall, while the recognition system shows promise, there is clear room for improvement in handling the rich linguistic diversity found in Indic texts.

Chapter 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

Scene Text Extraction from natural images is an important research domain with various applications such as socument digitaization, translation systems, inventory management system. Very little progress is done on different Indic Languages under this topic. The complexity of Indic Scripts pose unique challenges. This thesis explores the use of deep learning techniques, with a focus on Vision Transformers, to address these challenges in Scene Text Detection and Scene Text Recognition for Indic Scripts.

The proposed approach utilizes Vision Transformers to achieve robust detection and recognition of text in natural scenes. Experimental evaluations indicate the model achieves competitive performance. Comapred to baseline methods, we see noticeable improvements in F-scores in detection and CRR in recognition. The model shows signs in handling multilingual environment. Synthetic Data and Transfer Learning has played pivotal role in generalizing effectively of the model across different Indic Scripts.

5.2 Compliance

Scene Text Detection

- Develop a deep learning-based model capable of accurately identifying text regions in complex scene images.
  Fully Achieved - Using UperNet Architecure and extending it with DB-Net Methodology to localize the text regions. With use of Vision Trans-

former as it's backbone, results shows robust detection and competitive performance.

- Use different datasets to test and improve the overall quality of the model, including the use of a variety of Indic languages, fonts, and images.
  Fully Achieved - Datasets like SynthText, TotalText, IndicSTR12 are used to train the models, which consists of synthetic and real world images.

- Generate high-quality segmentation masks for further recognition processing.
  Fully Achieved - Based on the results, we've seen that it is able to generate high segmentation masks, and sometimes better than ground truth which consists of mask of a singular language.

- Evaluate the detection system on standard benchmarks using Precision, Recall and F-Score.
  Fully Achieved - The model is tested on standard metrics used in Detection tasks, which are Precision, Recall and F-Score.

B. Scene Text Recognition Objectives

- Design a recognition model that can transcribe cropped text images into machine-readable sequences with high accuracy, even under noise or partial occlusion.
  Fully Achieved - Combined pre-trained models of DeIT and IndicBART and finetuned it with Indic Texts. This approach is similar to the recent trend of Vision Encoder and Text Decoder.

- Improve robustness of character-level recognition by leveraging sequence-to-sequence learning and contextual language modeling.
  Paritally Achieved - Although there are promising results, certain languages have performed underwhelmingly. Languages such as Telugu, Tamil, Malayalam, etc.

- Support multi-lingual or script-agnostic recognition for better adaptability to various real-world settings.
  Paritally Achieved - Although the model is able to recognize multilingual text, it is robust, and performs poorly for some languages. The approach used requires the text to be transliterated to Devanagiri scripts

and trained on it. And in post-processing we can convert it back to it's original text based on the Language Token.

- Evaluate the recognition system on standard benchmarks using CER (Character Error Rate) and WER(Word Error Rate).
  Fully Achieved - The model is evaluated on standard metrics like CER and WER.

We can say that we have achieved all of our objectives for Detection. But for Recognition objectives there is still much room to improve for specific languages.

5.3 Limitations

Despite the promising results, the proposed methodology has limitations which allows for futher investigation on this topic.

- **Computational Complexity**: The high computational demands of transformer-based architectures require significant resources, which may not be feasible for low-power devices.
- **Multilingual Detection Challenges**: The model occasionally struggles to accurately detect text regions when multiple Indic scripts are present simultaneously in an image.
- **Recognition Inaccuracies**: Certain Indic scripts, particularly those with complex character sets or underrepresented in training data, exhibit lower recognition accuracy.

5.4 Future Work

To address the identified limitations and further advance the field of scene text extraction for Indic scripts, several directions for future research are proposed:

- **Enhanced Annotated Datasets**: Developing larger, more diverse, and well-annotated datasets for rare Indic scripts will improve model robustness and generalization.

- **Multi-Head Detection Architectures**: Implementing multi-head detection mechanisms, where each head is specialized for a specific Indic script, could enhance the model's ability to accurately localize text regions in multilingual scenes. Using the position of head to determine which language the identified text is, can be provided to Recognizer which could lead to better predictions.
- **Better Indic Script Recognition**: Transitioning from the current approach of converting Indic scripts to an intermediary Devanagari representation to direct, native recognition of each script's characters could reduce errors and improve efficiency.
- **Optimized Models for Resource-Constrained Devices**: Exploring model compression techniques, such as pruning or quantization, could enable the deployment of transformer-based models on low-resource devices without sacrificing performance
- **Knowledge Distillation for Lightweight Models**: Applying knowledge distillation to transfer the capabilities of large Vision Transformer models to smaller, computationally efficient architectures could facilitate on-device deployment while maintaining high accuracy

These future directions aim to enhance the robustness, efficiency, and applicability of scene text extraction systems for Indic scripts, paving the way for their widespread adoption in diverse, real-world scenarios.

BIBLIOGRAPHY

[1] Youngmin Baek, Bado Lee, Dongyoon Han, Sangdoo Yun, and Hwalsuk Lee. Character region awareness for text detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9365–9374, 2019.

[2] Darwin Bautista and Rowel Atienza. Scene text recognition with permuted autoregressive sequence models. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 178–196, 2022.

[3] Dongping Cao, Yong Zhong, Lishun Wang, Yilong He, and Jiachen Dang. Scene text detection in natural images: A review. *Symmetry*, 12(12), 2020.

[4] Jun Chen, Haiyang Yu, Jianqi Ma, Han Li, and Xianglin Yang. Vision permutable extractor for scene text recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1234–1243, 2023.

[5] Xiaoxue Chen, Lianwen Jin, Yuanzhi Zhu, Canjie Luo, and Tianwei Wang. Text recognition in the wild: A survey. *ACM Comput. Surv.*, 54(2), March 2021.

[6] Chee Kheng Ch'ng and Chee Seng Chan. Total-text: A comprehensive dataset for scene text detection and recognition. In *14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, pages 935–942, 2017.

[7] Raj Dabre, Himani Shrotriya, Anoop Kunchukuttan, Ratish Puduppully, Mitesh M Khapra, and Pratyush Kumar. Indicbart: A pre-trained model for indic natural language generation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1833–1849, 2022.

[8] Yongkun Du, Zhineng Chen, Caiyan Jia, Xiaoting Yin, Tianlun Zheng, Chenxia Li, Yuning Du, and Yu-Gang Jiang. Svtr: Scene text recognition with a single visual model. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 884–890, 2022.

[9] Ankush Gupta, Andrea Vedaldi, and Andrew Zisserman. Synthetic data for text localisation in natural images. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2315–2324, 2016.

[10] Mingxin Huang, Yuliang Liu, Zhenghao Peng, Chongyu Liu, Dahua Lin, Shenggao Zhu, Nicholas Yuan, Kai Ding, and Lianwen Jin. Swintextspotter: Scene text spotting via better synergy between text detection and text recognition. In *Proceedings of the*

IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 4593–4603, 2022.

[11] Mingxin Huang, Jiaxin Zhang, Dezhi Peng, Hao Lu, Can Huang, Yuliang Liu, Xiang Bai, and Lianwen Jin. Estextspotter: Towards better scene text spotting with explicit synergy in transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 19495–19505, 2023.

[12] Dimosthenis Karatzas et al. Icdar 2015 competition on robust reading. In *13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 1156–1160, 2015.

[13] Tauseef Khan, Ram Sarkar, and Ayatullah Faruk Mollah. Deep learning approaches to scene text detection: a comprehensive review. *Artificial Intelligence Review*, 54(5):3239–3298, Jun 2021.

[14] Minghao Li, Tengchao Xu, Lei Liu, Yong Chen, Hao Wang, Lei Yu, Dandan Huang, Yun Du, and Wenhu Yang. Trocr: Transformer-based optical character recognition with pre-trained models. *arXiv preprint arXiv:2109.10282*, 2021.

[15] Minghui Liao, Zhaoyi Wan, Cong Yao, Kai Chen, and Xiang Bai. Real-time scene text detection with differentiable binarization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(07):11474–11481, 2020.

[16] Hui Liu, Ming Liu, Dan Li, Yong Wang, and Yong Peng. Lightweight scene text recognition based on transformer. *Applied Sciences*, 13(4):2376, 2023.

[17] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10012–10022, 2021.

[18] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11976–11986, 2022.

[19] Shangbang Long, Xin He, and Cong Yao. Scene text detection and recognition: The deep learning era. *International Journal of Computer Vision*, 129(1):161–184, 2021.

[20] Shangbang Long, Jiaqiang Ruan, Wenjie Zhang, Xin He, Wenhao Wu, and Cong Yao. Textsnake: A flexible representation for detecting text of arbitrary shapes. In *Proc. European Conference on Computer Vision (ECCV)*, pages 20–36, 2018.

[21] Harsh Lunia, Ajoy Mondal, and C. V. Jawahar. *IndicSTR12: A Dataset for Indic Scene Text Recognition*, page 233–250. Springer Nature Switzerland, 2023.

[22] Minesh Mathew, Ajeet Singh, and C. Jawahar. Multilingual ocr for indic scripts. pages 186–191, 04 2016.

[23] Minesh Mathew, Ajeet Kumar Singh, and C. V. Jawahar. Benchmarking scene text recognition in devanagari, telugu and malayalam. In *Proc. 12th IAPR Workshop on Document Analysis Systems (DAS)*, pages 180–185, 2016.

[24] Anand Mishra, Karteek Alahari, and C. V. Jawahar. Scene text recognition using higher order language priors. In *British Machine Vision Conference (BMVC)*, 2012.

[25] Fatemeh Naiemi, Vahid Ghods, and Hassan Khalesi. Scene text detection and recognition: a survey. *Multimedia Tools and Applications*, 81(14):20255–20290, Jun 2022.

[26] Nibal Nayef et al. Icdar 2019 robust reading challenge on multi-lingual scene text detection and recognition. In *15th International Conference on Document Analysis and Recognition (ICDAR)*, pages 1582–1587, 2019.

[27] S. Saxena et al. Advances in scene text detection using deep learning architectures. *Computer Vision and Image Understanding*, 190:102845, 2020.

[28] A. Singh et al. Modular deep learning frameworks for scene text detection in indic scripts. *IEEE Transactions on Image Processing*, 30:5678–5690, 2021.

[29] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers and distillation through attention. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 10347–10357, 2021.

[30] Andreas Veit, Tomas Matera, Lukas Neumann, Jiri Matas, and Serge Belongie. Coco-text: Dataset and benchmark for text detection and recognition in natural images. *arXiv preprint arXiv:1601.07140*, 2016.

[31] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 418–434, 2018.

[32] MingKun Yang, Yushuo Guan, Minghui Liao, Xin Yang, Kaigui Liu, Jinwen Zhang, and Ming-Hsuan He. Spin: Structure-preserving inner offset network for scene text recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 3305–3313, 2021.

[33] Qixiang Ye and David Doermann. Text detection and recognition in imagery: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(7):1480–1500, 2015.

[34] Xiang Yue, Zhanghui Kuang, Chenhao Lin, Hongbin Sun, and Wayne Zhang. Robustscanner: Dynamically enhancing positional clues for robust text recognition. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 135–151, 2020.